

Empirical Performance Comparison of Optimization Algorithms for Machine Learning/Deep Learning

머신러닝/딥러닝을 위한 최적화 알고리즘의 성능비교

Kiu-Sik Kim . Duk-Sun Shim

김규식* . 심덕선†

Abstract

An optimization algorithm is essential for minimizing loss (or objective) functions in machine learning and deep learning. This article compares empirically the performance of 8 optimization algorithms which are used for training in machine learning and deep learning. The optimization algorithms considered in this article consist of 6 well-known algorithms such as SGD, SGD momentum, Nesterov momentum, AdaGrad, RMSprop, and Adam algorithms, and 2 recent algorithms such as KO and mSGD algorithms. Three kind of data is used for the performance comparison of optimization algorithms. First is the use of two functions, which have many local minima. Second is the use of MNIST data set which consists of handwritten digits from 0 to 9. The last one is the use of CIFAR-10 which consists of 10 kinds of images such as airplane, car, cat, dog, and so on. With the three kind of data set, seven cases are considered including FNN, CNN, and AE to compare the performance of 8 optimization algorithms.

Key Words

Deep learning, Machine learning, Optimization algorithm, SGD, SGD momentum, Nesterov momentum, AdaGrad, RMSprop, Adam, KO, mSGD

1. 서론

최근 들어 빅데이터의 사용과 빠른 데이터 처리가 가능해짐에 따라 인공지능망을 사용하는 딥러닝이 컴퓨터 비전과 자연어 처리 분야에서 큰 성과를 내고 있으며, 모델예측제어에도 딥러닝을 이용하는 연구가 최근 발표되었다[1]. 딥러닝뿐만 아니라 머신러닝에서도 시스템의 모델을 학습시키는 것이 필요한데, 학습은 주어진 데이터를 사용하여 모델의 가중치 파라미터 값을 구하는 과정을 의미한다. 주어진 문제에 따라 손실 함수를 정의하고 이 손실함수의 값이 작아지도록 모델의 파라미터를 훈련하는데 이 과정에서 최적화 알고리즘의 사용이 필수적이다.

머신러닝/딥러닝의 학습에 사용되는 최적화 알고리즘은 확률적 경사 하강 알고리즘(SGD, stochastic gradient decent)[2-4]을 기본으로 해서 이를 개선한 알고리즘들인 SGD momentum[2-5], Nesterov momentum[3], AdaGrad (adaptive gradient)[2-4, 6], AdaDelta[2-4, 7], AdaMax[3], RMSprop (root mean square propagation)[2-4], Adam (Adaptive momentum estimation)[2-4, 8], Nadam (Nesterov-accelerated Adam)[3],

mSGD(modified SGD)[9]가 있고 또 칼만필터를 이용한 KO (Kalman optimizer)[10] 등이 있다([11, 12] 참고).

제안된 최적화 알고리즘이 많아짐에 따라 여러 알고리즘의 성능을 비교하는 연구 결과도 발표[13-15] 되었는데, [13]에서는 SGD, SGD momentum, Adam의 세 알고리즘의 성능을 비교하였고, [14]에서는 SGD, SGD momentum, Nesterov momentum, RMSprop, Adam, Nadam의 여섯 알고리즘의 성능을 비교하였고, [15]에서는 SGD, AdaGrad, AdaDelta, Adam의 네 알고리즘에 대한 성능을 비교하였다.

본 논문에서는 잘 알려진 SGD, SGD momentum, Nesterov momentum, AdaGrad, RMSprop, Adam의 6개 알고리즘과 최근에 발표된 KO와 mSGD의 2개의 알고리즘의 성능을 실험적으로 비교해보고자 한다. 본 논문에서는 기연급한 8개의 최적화 알고리즘의 수렴 성능을 다양한 경우에서 비교하기 위해 지도 학습인 분류(classification)와 비지도 학습인 오토인코더(auto encoder, AE)를 사용하고, 데이터로는 0에서 9까지 손글씨체 데이터인 MNIST 데이터와 10개의 이미지 클래스 데이터인 CIFAR-10 데이터 셋을 사용하였고 또 함수도 사용한다. 구체적으로 다음과 같이 세가지 데이터에 대하여 비교해 본다. 첫

† Corresponding Author : School of Electrical and Electronics Engineering, Chung-Ang University, Republic of Korea.

E-mail : dshim@cau.ac.kr <https://orcid.org/0000-0003-0141-9498>

* School of Electrical and Electronic Engineering, Chung-Ang University, Republic of Korea.

E-mail : tlfjja519@naver.com <https://orcid.org/0009-0001-7642-5764>

Received : Jan. 16, 2025 Revised : May 24, 2025 Accepted : Jul. 10, 2025

째로 많은 지역 최솟값을 갖는 함수인 $f(x,y) = x^2 + y^2 + 5\sin(x+y^2)$ 에 대해서 최적화 알고리즘의 수렴 성능을 비교해 본다. 둘째로 MNIST 데이터를 사용하여 완전 연결된 순방향 신경망(forward neural network, FNN), 합성곱 신경망(convolutional neural network, CNN), 또 오토인코더(auto encoder, AE)를 사용할 때 최적화 알고리즘의 수렴 성능을 비교 해본다. 셋째로 CIFAR-10 데이터 셋을 사용하여 CNN을 사용 할 때 최적화 알고리즘의 수렴 성능을 비교해 본다. 수렴 성능 외에도 8개의 최적화 알고리즘 별 계산 시간을 비교하고, 분류 문제에 대해서는 분류 성능을 비교한다.

2. 머신러닝/딥러닝에 사용되는 최적화 알고리즘

2.1 최적화 알고리즘의 개념

이 절에서는 여러 최적화 알고리즘들의 개념을 알아보겠다. SGD는 신경망의 가중치 값에서 손실함수의 경사값과 학습률(learning rate, lr)의 곱을 빼서 가중치 값을 갱신하는 가장 기본적인 최적화 알고리즘이다. 이 SGD의 성능개선을 위해서 현재의 가장 가파른 손실함수의 경사 방향과 기존의 진행하던 경사 방향인 관성을 같이 사용한 방법이 SGD momentum이다. Nesterov momentum은 SGD momentum을 개선한 것으로 현재 시점의 경사도를 사용하는 대신에 한 스텝 후의 경사도를 사용한 점이 다르다. AdaGrad는 손실함수의 학습률을 적응적으로 정하는 알고리즘으로, 기울기의 변화가 크면 작은 폭으로 이동하고 작으면 큰 폭으로 이동하도록 조정한다. AdaGrad는 손실함수의 경사가 큰 상태에서 시작한 경우에는 학습 초반에 학습이 중단되는 단점이 있는데 이를 개선한 알고리즘이 RMSprop이다. SGD momentum과 RMSprop을 결합한 알고리즘이 Adam으로 최적화 진행 방향에 관성을 주고 경사의 변화량에 따라 적응적 학습률을 갖도록 한 알고리즘이다. SGD 알고리즘에서 학습률은 하이퍼 파라미터로 탐색을 통해서 미리 정하는 상수 값인데, 이 학습률을 고정하지 않고 가중치 갱신시마다 여러 개를 랜덤하게 발생시켜서 손실함수의 값이 가장 작게 나오는 학습률을 선택하여 사용하도록 개선시킨 것이 mSGD 알고리즘이다. k개의 랜덤 학습률을 사용하는 것을 k-point mSGD 알고리즘이라고 한다[9]. KO는 칼만필터를 사용하는 알고리즘으로서 상태 방정식을 사용하므로 상태 방정식의 계수들과 상태식과 측정식의 노이즈에 대한 공분산이 학습률 외에 추가적인 파라미터가 된다.

2.2. 신경망의 구현 및 최적화 알고리즘의 파라미터

본 연구에서는 기존의 잘 알려진 SGD, SGD momentum, Nesterov momentum, AdaGrad, RMSprop, Adam의 6개 알고리즘은[12, <https://github.com/WegraLee/deep-learning-from-scratch>]에서 제공하는 클래스를 사용하였고 최근에 발표된 KO와 mSGD는 직접 코딩하여 사용하였다. 또 신경망에 사용된 FNN과 CNN은 [12]에서 제공한 기본 틀을 사용하였고, 프로그램은 파이썬 언어를 사용하였다. 최적화 알고리즘의 학습률

(lr)은, 표 1의 $f_1(x,y)$ 함수의 경우에는 각각의 알고리즘에 적합한 값을 탐색하여 사용하였고, $f_2(x,y)$ 함수의 경우에는 최적화 알고리즘의 민감도를 알아보기 위해 $f_1(x,y)$ 함수를 위해서 탐색한 값을 그대로 사용하였다. 4장 이후의 FNN, CNN, AE의 경우에는 최적화 알고리즘의 학습률(lr)은 기본값(default)에 10 또는 0.1의 배수를 곱하며 적합한 값을 찾아 사용하였고, 구체적인 값들은 해당 절에서 제시하였다. 최적화 알고리즘의 다른 파라미터들은 기본값(default)을 그대로 사용하였다. 최적화 알고리즘에서 활성화 함수는 ReLU를 사용하였고, 가중치의 초기값은, FNN은 He 초기값을, CNN과 AE는 표준편차가 0.01의 정규분포를 사용하였다.

3장부터는 용어의 간략성을 위해 SGD momentum은 Momentum(또는 MMT)으로 사용하고 Nesterov momentum은 Nesterov로 사용한다.

3. 다수의 지역 최저점을 갖는 함수에 대한 최적화 알고리즘의 수렴성능

본 장에서는 다수의 지역 최저점을 갖고 있으며, 전역 최저점을 알고 있는 함수인 $f(x,y) = x^2 + y^2 + 5\sin(x+y^2)$ 에 대해서 2장에서 언급한 8개의 최적화 알고리즘의 수렴성능을 분석한다. 그림 1은 함수 $f(x,y)$ 의 높이를 색으로 보여 주는데 파랑색이 진할수록 함수의 값이 작다. 그림 1에서 보는 것처럼 이 함수는 다수의 지역 최저점을 갖고 있으며 전역 최저점은 (-1.11, 0) 근처에 있다. 또 지역 최저점을 둘러싸는 등고선은 전역 최저점에서 멀어질수록 가늘고 길어지는 것을 볼 수 있다.

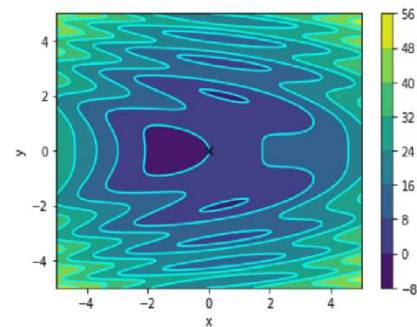


그림 1 $f(x,y) = x^2 + y^2 + 5\sin(x+y^2)$ 의 등고선도
Fig. 1 Contour plot of $f(x,y) = x^2 + y^2 + 5\sin(x+y^2)$.

최적화 알고리즘의 수렴 성능은 초기 위치에 따라 달라질 수 있으므로, 초기 위치를 (5, 5), (-2, 4), (3, 4), (-5, 3)로 변경하면서 수렴 성능을 살펴보았다. SGD, Nesterov, AdaGrad, KO, mSGD의 다섯 알고리즘은 초기 위치 네 곳 모두에서 전역 최저점으로 잘 수렴하였고, RMSprop은 두 곳에서 전역 최저점으로 수렴, 두 곳에서 지역 최저점으로 수렴하였고, Adam은 세 곳에서 전역 최저점으로 수렴, 한 곳에서 지역 최저점으로 수렴하였고, Momentum은 한 곳에서 전역 최저점으로 수렴,

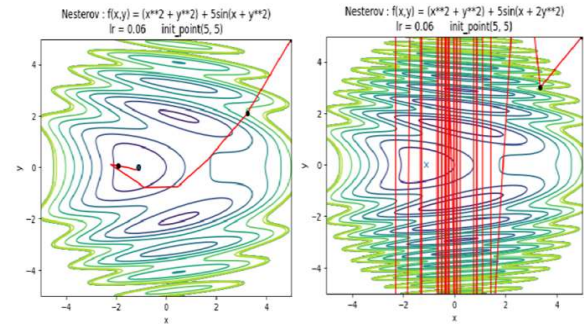
두 곳에서 지역 최저점으로 수렴, 한 곳에서는 발산하였다.

본 논문에서는 8개의 최적화 알고리즘이 모두 전역 최저점으로 수렴하는 (5, 5)를 초기점으로 사용하였고, 그림 2에서 수렴 경로를 보여준다. 그림 2에서 좌측의 그림은 함수 $f(x,y)$ 에 대한 그림이고, 우측은 $f(x,y)$ 에서 사인(sine)함수 내의 y^2 대신에 $2y^2$ 을 사용한 그림이다. 이 두 함수를 구분하기 위하여 식 (1)과 같이 $f_1(x,y)$ 과 $f_2(x,y)$ 로 구분하여 사용하기로 한다.

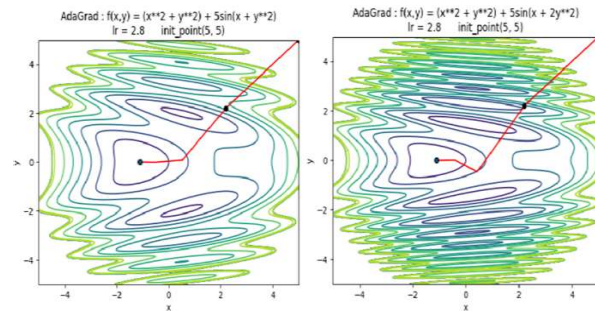
$$f_1(x,y) = x^2 + y^2 + 5\sin(x+y^2) \quad (1-1)$$

$$f_2(x,y) = x^2 + y^2 + 5\sin(x+2y^2) \quad (1-2)$$

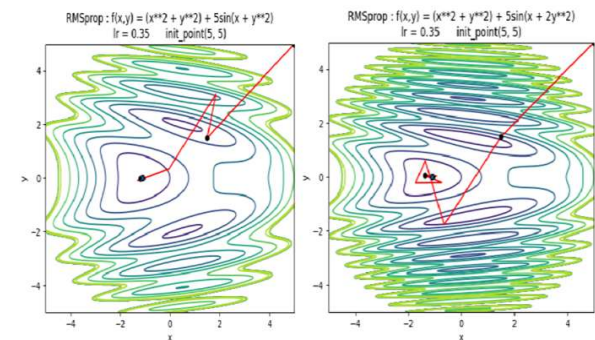
함수 $f_2(x,y)$ 를 사용한 이유는 최적화 알고리즘의 민감도를 살펴보기 위함이다. 그림 2에서 보는 바와 같이 f_2 의 등고선은 f_1 의 등고선 보다 y 축 방향으로 압축되어 최적화 알고리즘이 전역 최저점으로 수렴하기 어렵다. 그림 2의 좌측 그림은 f_1 함수에 대해 하이퍼 파라미터인 학습률 lr 을 최적화 알고리즘 별로 성능이 좋은 값을 탐색하여 정했고(표 1 참조), 그림 2의 우측 그림은 각각의 최적화 알고리즘의 강인성 또는 민감도를 살펴볼 수 있도록 f_1 함수에 대해서 구한 학습률을 f_2 에 그대로 사용하였다. 따라서 그림 2의 우측 그림들은 대부분의 최적화 알고리즘의 수렴 성능이 좌측 그림에 비해서 떨어지는 것을 볼 수 있는데, (b) Momentum 알고리즘, (c) Nesterov 알고리즘, (g) KO 알고리즘은 수렴하지 못하고 진동하는 것을 볼 수 있다. 그림 2에서 도트(●)는 시작점 (5, 5)에서 5번의 학습 iteration마다 한 번씩 사용하였다.



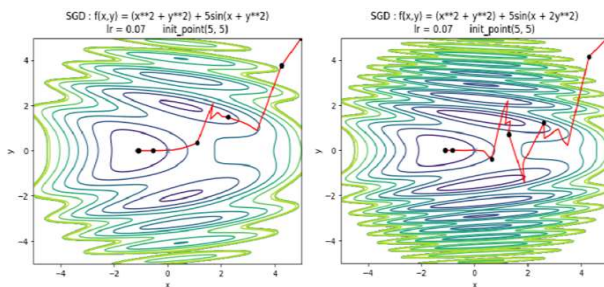
(c) Nesterov 알고리즘



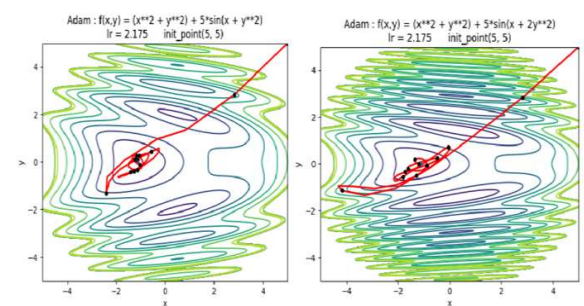
(d) AdaGrad 알고리즘



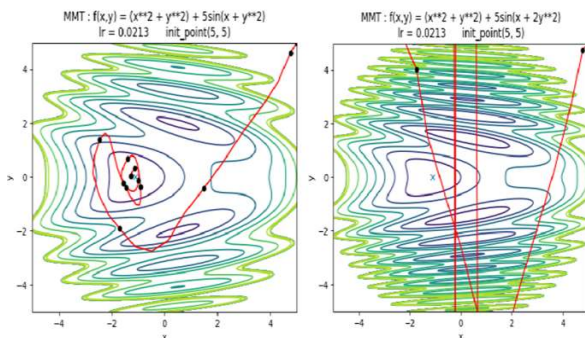
(e) RMSprop 알고리즘



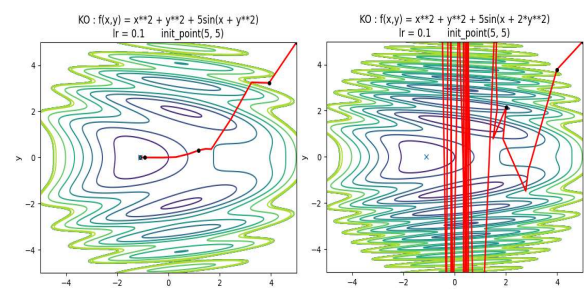
(a) SGD 알고리즘



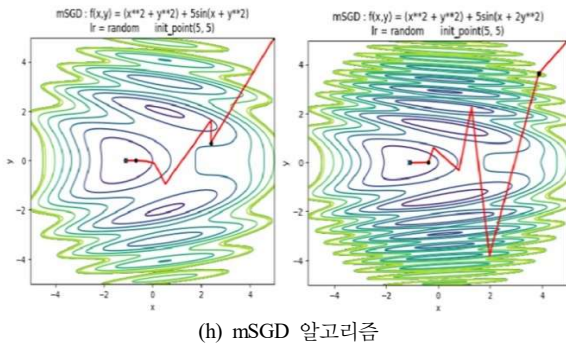
(f) Adam 알고리즘



(b) Momentum 알고리즘



(g) KO 알고리즘



(h) mSGD 알고리즘

그림 2 함수 $f_1(x,y)$ (좌측 그림)과 $f_2(x,y)$ (우측 그림)에 대한 8개의 최적화 알고리즘의 수렴 궤적

Fig. 2 Convergence trajectory of 8 optimization algorithms for functions $f_1(x,y)$ (left plot) and $f_2(x,y)$ (right plot)

표 1은 f_1, f_2 함수에 사용한 8개의 최적화 알고리즘의 학습률 lr 과 수렴 성능의 순위(rank)로 보여주며, 수렴 성능은 매 iteration 마다 현재 위치에서 전역 최저점 $(-1.11, 0)$ 까지의 거리를 더한 값, 즉 오차 거리(e)의 1-norm 값을 성능 지표로 사용하였다. 오차 거리(e)의 1-norm은 $\|e\|_1 = \sum_{n=1}^N |e_n|$ 이다.

4장 이후에 MNIST 데이터와 CIFAR-10 데이터를 사용한 경우에는 교차 엔트로피 오차를 손실함수로 사용한다.

표 1에서 KO 알고리즘에 사용한 칼만필터의 상태식은 식 (2)와 같다.

$$x_k = ax_{k-1} + w_{k-1}, \quad y_k = cx_k + v_k \quad (2)$$

여기서 $w_k \sim \mathcal{N}(0, Q)$, $v_k \sim \mathcal{N}(0, R)$

식 (2)에서 x_k 는 상태변수, y_k 는 측정값, w_k 는 플랜트 잡음, v_k 는 측정치 잡음, Q 는 w_k 의 오차 공분산, R 은 v_k 의 오차 공분산이다.

식 (2)에서 $a=1$, $c=1$, $Q=1$, $R=1$ 의 값을 사용하였다. 이 값들은 이후 FNN, CNN과 AE에도 같은 값이 적용된다.

표 1 함수 f_1 과 f_2 에 대한 8개의 최적화 알고리즘의 수렴 궤적

Table 1 Convergence ranking of 8 optimization algorithms for functions f_1 and f_2

optimizer	$f_1(x,y)$		$f_2(x,y)$		lr
	$\ e\ _1$	ranking	$\ e\ _1$	ranking	
SGD	49.816	7	60.339	5	0.07
Momentum	71.227	8	8499.4	6	0.0213
Nesterov	12.853	3	Inf	8	0.06
AdaGrad	6.2728	1	6.3632	1	2.8
RMSprop	9.0897	2	7.4916	2	0.35
Adam	31.868	6	46.239	4	2.175
KO	27.942	5	22756.	7	0.1
mSGD	13.216	4	16.567	3	uniformly random in [0, 0.3]

mSGD 알고리즘의 경우에는 $[0, 0.3]$ 구간에서 3개의 uniform한 랜덤 값(3-point mSGD[9])을 발생시켜서 사용하였고, 앞으로도 계속해서 3-point mSGD를 사용한다.

4. FNN 사용시 최적화 알고리즘의 수렴성능

4.1 MNIST 데이터 셋

MNIST 데이터는 0에서 9까지 10개의 손글씨 숫자의 이미지 셋으로, 훈련 이미지가 60,000장, 시험 이미지가 10,000장으로 구성되어 있다. 각각의 숫자 데이터는 28×28 의 회색조 이미지(1채널)이고 각 픽셀은 0에서 255까지의 값을 갖는다. 그림 3은 MNIST 데이터의 일부를 보여주고 있는데 각각의 숫자 이미지에 실제 숫자가 레이블로 붙어 있다.



그림 3 MNIST 데이터 셋

Fig. 3 MNIST Data Set

4.2 MNIST 데이터 셋에 FNN 사용시 최적화 알고리즘의 파라미터 값

본 훈련에서 사용한 최적화 알고리즘의 파라미터는 다음과 같다. SGD는 $lr = 0.1$, Momentum은 $lr = 0.1$, momentum = 0.9, Nesterov는 $lr = 0.1$, momentum = 0.9, AdaGrad는 $lr = 0.1$, RMSprop은 $lr = 0.01$, decay_rate = 0.99, Adam은 $lr = 0.01$, beta1 = 0.9, beta2 = 0.999, KO는 $lr = 1$, mSGD는 $lr \in [0, 0.3]$ 이고 uniform한 확률 밀도를 갖는 랜덤변수다.

4.3 MNIST 데이터 셋에 FNN 사용시 최적화 알고리즘의 수렴성능

MNIST의 각 숫자 데이터는 28×28 의 행렬로 표현되는 이미지인데 이를 행렬에서 784×1 의 벡터로 변형시켜 FNN에 사용한다. 이 절에서는 FNN을 훈련시키는 과정에서 8개의 최적화 알고리즘들의 수렴 성능을 비교해 본다. 훈련 이미지 60,000장에 대해서 미니배치 크기는 128을 사용한다.

그림 4와 그림 5는 각각 2-layer와 4-layer FNN에 대한 최적화 알고리즘의 수렴 성능을 보여주고 있는데, 2-layer와 4-layer를 사용한 것은 얇은 신경망(2-layer)과 깊은 신경망(4-layer)에서의 수렴 성능을 비교해 보기 위함이다. 그림 4와 그림 5의 아래 그림은 수렴 성능을 비교해 보기 쉽도록 50-point 이동 평균값을 보여 준다.

최적화 알고리즘의 수렴 성능은 학습이 끝났을 때의 손실 함수값을 비교하면 되는데, 매 iteration마다 128개의 미니배치

데이터를 무작위로 선택해서 사용하기 때문에, 그림 4와 그림 5에서 보는 바와 같이 알고리즘별로 손실 함숫값이 일정하게 수렴하지 않고 변동(fluctuation)이 생긴다. 따라서 손실 함숫값의 변동을 고려하여 최적화 알고리즘의 수렴 성능 지표는 iteration의 마지막 5%에 대한 손실 함숫값의 1-norm으로 하였다.

표 2는 그림 4와 그림 5에서의 8개의 최적화 알고리즘의 수렴 결과와 계산 시간을 보여 준다. $\|z\|_1^{5\%}$ 는 손실함수 값의 마지막 5%의 구간에 대해 1-norm을 계산한 것을 의미하며, 이와 같은 수렴성능 지표는 이후 표 3~4에서도 같은 방법을 사용한다. 2-layer의 경우에는 RMSprop이 1순위, KO가 2순위를 보여주고 있으며 4-layer의 경우에는 Nesterov가 1순위, mSGD가 2순위를 보여주고 있다.

표 2에서 시간은, 최적화 알고리즘 별로 1,000 iteration 동안의 계산 시간을 의미한다. 가장 간단한 알고리즘인 SGD의 학습 소요시간에 비해서 SGD를 개선한 알고리즘인 3-point mSGD 알고리즘의 소요시간은 SGD보다 2-layer일 때 62.6%, 4-layer일 때 50% 긴 것을 알 수 있다. 이 계산에 사용한 컴퓨터는 갤럭시 북4 울트라 노트북이고, 사양은 울트라7 프로세서(up to 4.8GHz), 32GB램, 엔비디아 지포스 GPU 6GB, 웨스턴 디지털 1TB SSD 등이다.

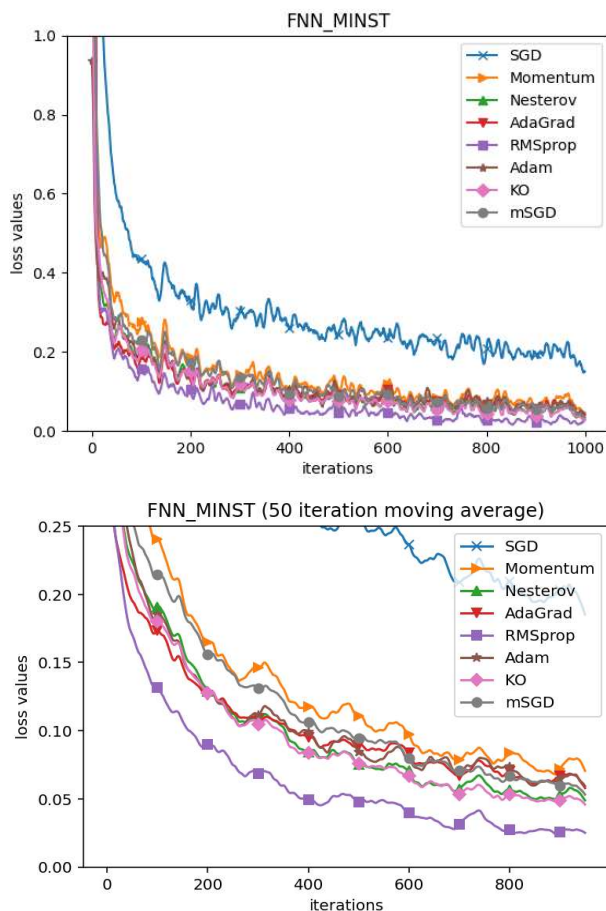


그림 4 MNIST 데이터, 2-layer FNN에 대한 8개의 최적화 알고리즘의 수렴 성능 (아래 그림은 50-point 이동평균)

Fig. 4 Performance of 8 optimization algorithms for MNIST data and 2-layer FNN (lower figure : 50-point moving average)

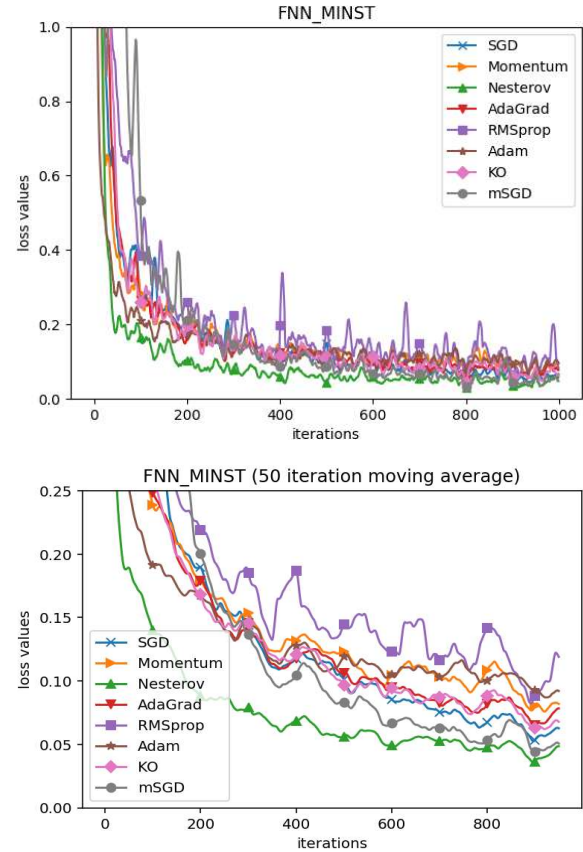


그림 5 MNIST 데이터, 4-layer FNN에 대한 8개의 최적화 알고리즘의 수렴 성능 (아래 그림은 50-point 이동평균)

Fig. 5 Performance of 8 optimization algorithms for MNIST data and 4-layer FNN (lower figure : 50-point moving average)

표 2 MNIST 데이터 사용시에 FNN에 대한 8개의 최적화 알고리즘의 수렴 성능 ($\|z\|_1^{5\%}$ 에서 5%는 iteration의 마지막 5%를 의미함)

Table 2 Convergence ranking of 8 optimization algorithms for FNN with MNIST data (5% in $\|z\|_1^{5\%}$ means the final 5% of iteration)

optimizer	FNN MNIST					
	2-layer			4-layer		
	$\ z\ _1^{5\%}$	ranking	time[s]	$\ z\ _1^{5\%}$	ranking	time[s]
SGD	9.2485	8	2.62	3.0819	3	4.48
Momentum	3.5305	7	2.76	3.9950	6	4.33
Nesterov	2.4250	3	2.80	2.3940	1	4.50
AdaGrad	2.8957	5	2.84	3.8964	5	4.39
RMSprop	1.2558	1	2.85	5.8485	8	4.40
Adam	2.9464	6	3.08	4.5262	7	4.68
KO	2.2829	2	3.24	3.3339	4	5.00
mSGD	2.6630	4	4.26	2.5176	2	6.72

5. CNN 사용시 최적화 알고리즘의 수렴성능

5.1 MNIST 데이터 셋에 CNN 사용시 최적화 알고리즘의 파라미터 값

본 훈련에서 사용한 최적화 알고리즘의 파라미터는 다음과

같다. SGD는 $lr = 0.1$, Momentum은 $lr = 0.1$, $momentum = 0.9$, Nesterov는 $lr = 0.1$, $momentum = 0.9$, AdaGrad는 $lr = 0.1$, RMSprop은 $lr = 0.001$, $decay_rate = 0.99$, Adam은 $lr = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, KO는 $lr = 1$, mSGD는 $lr \in [0, 0.3]$ 이고 uniform한 확률 밀도를 갖는 랜덤변수다. 필터 개수는 30개, 필터 크기는 3, 패딩 0, 스트라이드는 1을 사용한다.

5.2 MNIST 데이터 셋에 CNN 사용시 최적화 알고리즘의 수렴성능

MNIST 데이터 셋 사용시에 3-layer CNN에 대한 최적화 알고리즘의 수렴 성능은 그림 6에 있으며, 그림 6의 두번째 그림은 50-point 이동평균 값을 보여준다. 표 3에서는 최적화 알고리즘별로 수렴 순위를 보여준다. AdaGrad는 초반 수렴은 좋았으나 300 iteration 부근부터 수렴이 멈춰서 8순위의 수렴 성능을 보여준다. AdaGrad는 손실함수의 초기 경사가 큰 경우에 학습 초반에 학습이 중단되는 단점이 있는데 이 경우를 그것을 보여 주었다. Nesterov, KO, mSGD, Momentum, RMSprop이 근소한 차이로 1순위~5순위를 보여주고 있고, SGD와 Adam이 근소한 차이로 6, 7위를 보여주고 있다. 차이가 아주

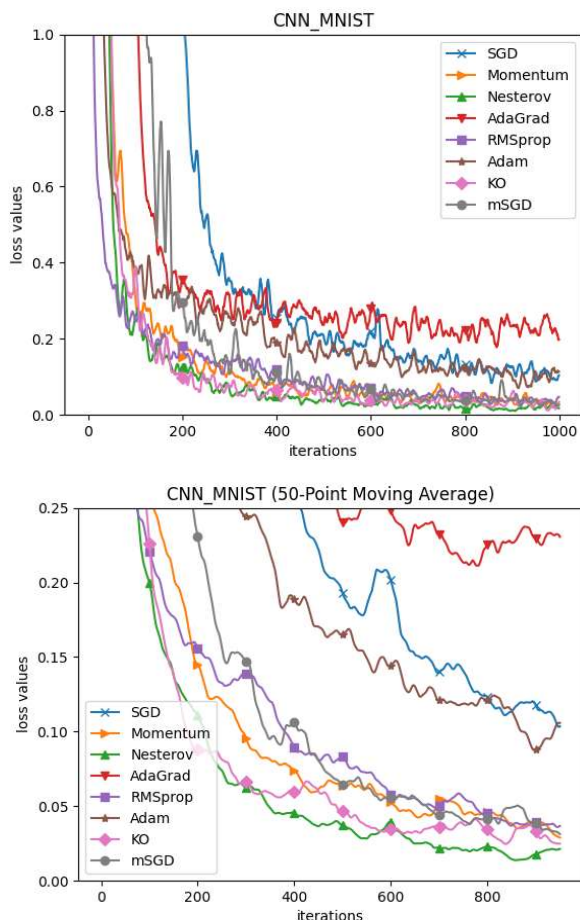


그림 6 MNIST 데이터, 3-layer CNN에 대한 8개의 최적화 알고리즘의 수렴 성능 (아래 그림은 50-point 이동평균)

Fig. 6 Performance of 8 optimization algorithms for MNIST data and 3-layer CNN (lower figure: 50-point moving average)

근소한 경우에는 실험에 따라 순위가 바뀔수도 있음을 고려해야 한다.

CNN MNIST의 경우 mSGD 알고리즘은 계산시간이 SGD 알고리즘에 비해서 7.3배 정도 더 걸리는 것을 볼 수 있다.

5.3 CIFAR-10 데이터 셋

CIFAR-10 데이터 셋은 10개의 다른 클래스(비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭)의 데이터 셋이다. 각 클래스가 칼라 이미지(3채널) 6,000개씩으로 구성되어 있어 모두 60,000개의 데이터 셋이며 훈련 이미지가 50,000장, 시험 이미지가 10,000장으로 구성되어 있다.

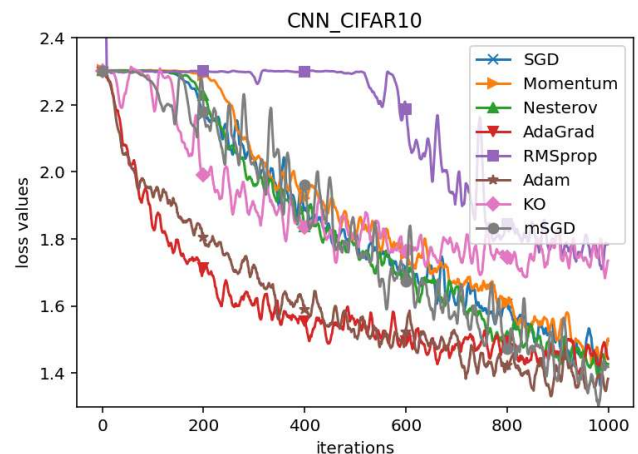
5.4 CIFAR-10 데이터 셋에 CNN 사용시 최적화 알고리즘의 파라미터 값

신경망의 훈련에 사용한 최적화 알고리즘의 파라미터는 다음과 같다. SGD는 $lr=0.1$, Momentum은 $lr=0.01$, $momentum=0.9$, Nesterov는 $lr=0.01$, $momentum=0.9$, AdaGrad는 $lr=0.01$, RMSprop는 $lr=0.01$, $decay_rate=0.99$, Adam은 $lr=0.001$, $\beta_1=0.9$, $\beta_2=0.999$, KO는 $lr = 1$, mSGD는 $lr \in [0, 0.1]$ 이고 uniform한 확률 밀도를 갖는 랜덤변수다. 필터 개수는 30개, 필터 크기는 3, 패딩 0, 스트라이드는 1을 사용한다.

5.5 CIFAR-10 데이터 셋에 CNN 사용시 최적화 알고리즘의 수렴성능

CIFAR-10 데이터 사용시에 3-layer CNN에 대한 최적화 알고리즘의 수렴 성능은 그림 7에서 보여 주는데, 그림 6의 CNN MNIST의 경우보다 8개 알고리즘 모두 수렴 속도가 상당히 늦은 것을 볼 수 있다. 표 3에서는 최적화 알고리즘별로 수렴 순위를 보여주는데 CNN MNIST의 경우에는 Nesterov 알고리즘이 1순위, KO가 2순위 반면에 CNN CIFAR-10의 경우에는 근소한 차이로 mSGD 알고리즘이 1순위, Adam은 2순위인 것을 볼 수 있다. mSGD 알고리즘은 200 iteration까지는 수렴이 늦었으나 그 후 빠르게 수렴하여 1순위를 보여주었다.

CNN CIFAR-10의 경우 mSGD 알고리즘은 계산시간이 SGD 알고리즘에 비해서 7.6배 정도 더 걸리는 것을 볼 수 있다.



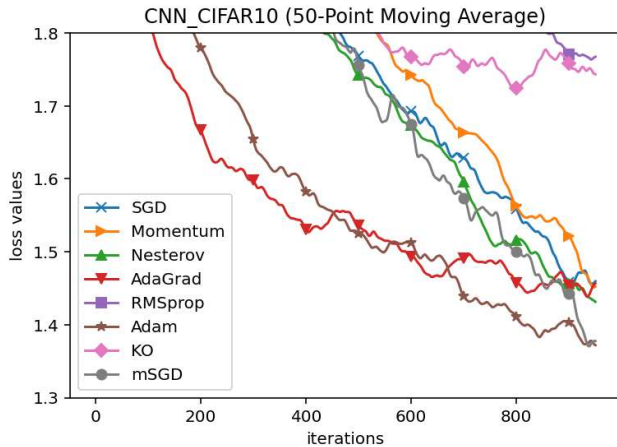


그림 7 CIFAR-10 데이터, 3-layer CNN에 대한 8개의 최적화 알고리즘의 수렴 성능 (아래 그림은 50-point 이동평균)

Fig. 7 Performance of 8 optimization algorithms for CIFAR-10 data and 2-layer CNN (lower figure: 50-point moving average)

표 3 MNIST 데이터와 CIFAR-10 데이터 사용시에 CNN에 대한 8개의 최적화 알고리즘의 수렴 성능 ($\|z\|_1^{5\%}$ 에서 5%는 iteration의 마지막 5%를 의미함)

Table 3 Convergence ranking of 8 optimization algorithms for CNN with MNIST and CIFAR-10 data (5% in $\|z\|_1^{5\%}$ means the final 5% of iteration)

optimizer	CNN MNIST			CNN CIFAR-10		
	3-layer			3-layer		
	$\ z\ _1^{5\%}$	ranking	time[s]	$\ z\ _1^{5\%}$	ranking	time[s]
SGD	5.1003	6	197.71	71.522	6	297.23
Momentum	1.4202	4	196.63	71.132	4	298.75
Nesterov	1.0634	1	197.01	70.182	3	300.15
AdaGrad	11.329	8	193.88	71.394	5	298.14
RMSprop	1.8014	5	201.19	86.632	8	296.58
Adam	5.1600	7	203.31	67.215	2	308.41
KO	1.2250	2	204.44	85.420	7	309.78
mSGD	1.3661	3	1443.0	66.791	1	2245.7

6. AE 사용시 최적화 알고리즘의 수렴성능

6.1 사용한 오토인코더 구조

오토인코더는 신경망을 사용하여 데이터의 압축을 자동으로 학습하는 모델이다. 입력 데이터는 인코더 신경망을 통하여 압축된 잠재변수가 되고 디코더 신경망을 통하여 출력 데이터를 만든다. 이 입력 데이터와 출력 데이터의 차를 가지고 손실함수를 만들기 때문에 오토인코더는 비지도학습으로 분류된다.

6.2 MNIST 데이터 셋에 오토인코더 사용시 최적화 알고리즘의 파라미터 값

신경망의 훈련에 사용한 최적화 알고리즘의 파라미터는 다음과 같다. SGD는 $lr=0.1$, Momentum는 $lr=0.01$, momentum=0.9, Nesterov는 $lr=0.01$, momentum=0.9, AdaGrad는

$lr=0.01$, RMSprop는 $lr=0.001$, decay_rate=0.99, Adam은 $lr=0.001$, beta1=0.9, beta2=0.999, KO는 $lr=1$, mSGD는 $lr \in [0, 0.1]$ 이고 uniform한 확률 밀도를 갖는 랜덤변수다.

6.3 MNIST 데이터 셋에 오토인코더 사용시 최적화 알고리즘의 수렴성능

오토인코더에 대한 최적화 알고리즘의 수렴 성능은 그림 8과 같이 초반에 빠른 수렴을 보여주며 RMSprop이 1순위, mSGD가 2순위임을 보여 준다. 그림 9에서는 최적화 알고리즘

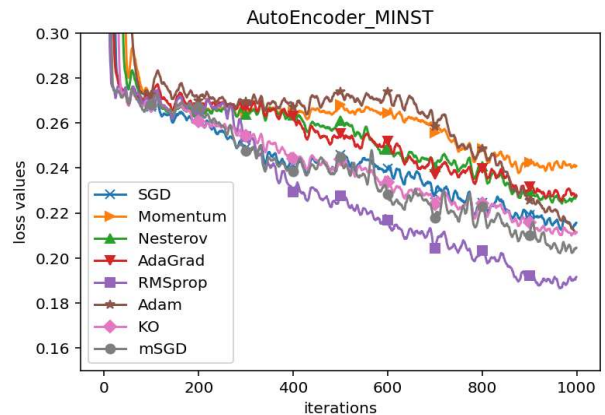


그림 8 MNIST 데이터 사용시 FNN 오토인코더에 대한 8개의 최적화 알고리즘의 수렴성능 (아래 그림은 50-point 이동평균)

Fig. 8 Performance of 8 optimization algorithms for MNIST data and FNN Auto Encoder (lower figure: 50-point moving average)

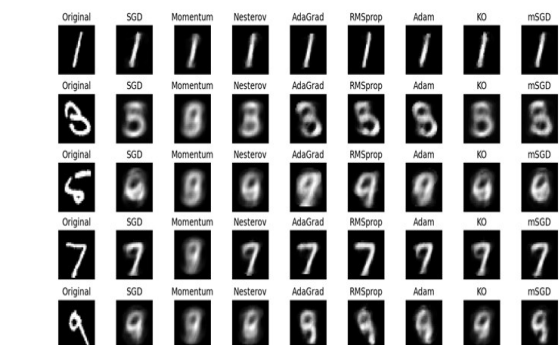


그림 9 MNIST 데이터 사용시 8개의 최적화 알고리즘에 대한 오토인코더의 입출력

Fig. 9 Inputs and outputs of auto encoder for 8 optimization algorithms

별로 MNIST 데이터 중 홀수에 대한 오토인코더의 입력과 출력을 보여준다.

표 4는 각 최적화 알고리즘의 수렴 순위와 계산 시간을 보여준다. 표 2와 표 3에서 FNN_MNIST, CNN_MNIST, CNN_CIFAR-10 모두 mSGD의 계산 시간이 제일 길었는데 AE의 경우 KO의 계산 시간이 제일 긴 것을 알 수 있다. mSGD의 수렴시간은 SGD에 비해서 68% 더 소요되었다.

표 4 MNIST 데이터 사용시에 오토인코더에 대한 8개의 최적화 알고리즘의 수렴 성능 ($\|z\|_1^{5\%}$ 에서 5%는 iteration의 마지막 5%를 의미함)

Table 4 Convergence ranking of 8 optimization algorithms for Auto Encoder with MNIST data (5% in $\|z\|_1^{5\%}$ means the final 5% of iteration)

optimizer	Auto Encoder		
	FNN		
	$\ z\ _1^{5\%}$	ranking	time[s]
SGD	10.551	4	13.4
Momentum	11.835	8	14.7
Nesterov	11.084	6	15.0
AdaGrad	11.207	7	16.4
RMSprop	9.2892	1	17.9
Adam	10.618	5	21.1
KO	10.381	3	27.3
mSGD	10.026	2	22.5

7. 최적화 알고리즘의 수렴성능 종합 및 분류 성능

7.1 8개의 최적화 알고리즘의 종합 수렴 성능

지금까지 표 2~4를 통해서 모두 7가지의 경우에 대해서 8개 최적화 알고리즘의 수렴 성능을 비교해 보았다. 수렴값의 차이가 미세한 경우에는 미니 배치의 특성상 프로그램을 수행할 때마다 순위가 다르게 나올 수도 있다. 따라서 여러 경우에

표 5 최적화 알고리즘의 종합적인 수렴 성능

Table 5 Total convergence ranking of optimization algorithms

optimizer	Total (7 cases) (Table 1~Table 4)		Total (5 cases) (Table 2~Table 4)	
	average ranking	ranking	average ranking	ranking
SGD	5.6	7	5.4	5
Momentum	6.1	8	5.8	7
Nesterov	3.6	2	2.8	2
AdaGrad	4.6	5	6	8
RMSprop	3.9	3	4.6	4
Adam	5.3	6	5.4	5
KO	4.3	4	3.6	3
mSGD	2.7	1	2.4	1

대한 평균 순위를 구해보는 것이 의미가 있다고 본다. 함수 2개를 포함한 전체 7가지 경우의 평균 순위와, 함수를 제외하고 실데이터만 사용한 5가지 경우의 평균 순위를 각각 계산하여 표 5에 정리하였다.

표 5의 좌측열을 보면 표 1~4의 7개 경우(함수 포함)의 평균 수렴 순위를 보여주고 있으며, 우측열은 표 2~4의 5개 경우(함수 제외)의 평균 수렴 순위를 보여주고 있다. 종합적으로 볼 때 mSGD가 알고리즘 계산 시간은 제일 오래 걸리지만 수렴 성능은 1순위를 보여주고 있다.

(본 논문에서 계산에 사용된 코드는 아래 링크 참조)

<https://github.com/superbigtrash/Empirical-Performance-Comparison-of-Optimization-Algorithms-for-Machine-Learning-Deep-Learning/tree/main>

7.2 8개의 최적화 알고리즘의 분류 성능

본 절에서는 MNIST 데이터 셋과 CIFAR-10 데이터 셋에 FNN과 CNN 사용시 8개 최적화 알고리즘에 대한 분류 성능을 비교해 본다. 그림 4~그림 7에서 1,000 iteration 동안 MNIST 데이터와 CIFAR-10 데이터를 훈련한 후에 각각 10,000개의 테스트 이미지를 사용하여 입력 이미지와 레이블을 비교하여 분류를 제대로 수행했는지의 성능을 표 6에 기재하였다. 또 수렴 성능과 비교해 볼 수 있도록 표 2와 표 3에 기재한 8개 최적화 알고리즘에 대한 수렴 성능을 표 7에 정리하였다. 표 6~7을 보면 손실함수의 경사만을 사용한 SGD, AdaGrad, RMSprop에 비해서 경사와 관성을 같이 사용하는 Momentum, Nesterov, Adam이 수렴성능과 분류성능 모두 우수한 것을 볼 수 있다.

표 6 MNIST 및 CIFAR-10 데이터 사용시에 8개 최적화 알고리즘의 분류 성능

Table 6 Classification performance of 8 optimization algorithms for MNIST and CIFAR-10 data

optimizer	FNN MNIST		CNN		Total (4 cases)	
	2-layer	4-layer	MNIST	CIFAR-10		
	test[%] (ranking)	test[%] (ranking)	test[%] (ranking)	test[%] (ranking)	average ranking	ranking
SGD	93.53 (8)	95.38 (7)	94.94 (8)	43.91 (5)	7.0	7
Momentum	97.04 (2)	96.35 (2)	97.95 (2)	45.64 (3)	2.3	2
Nesterov	97.21 (1)	96.65 (1)	98.02 (1)	46.77 (2)	1.3	1
AdaGrad	96.31 (6)	95.61 (6)	95.45 (7)	41.39 (6)	6.3	6
RMSprop	95.95 (7)	93.26 (8)	96.65 (6)	29.68 (8)	7.3	8
Adam	96.97 (3)	96.07 (4)	97.86 (3)	48.69 (1)	2.8	3
KO	96.77 (4)	95.88 (5)	97.46 (4)	35.39 (7)	5.0	5
mSGD	96.66 (5)	96.29 (3)	96.86 (5)	45.02 (4)	4.3	4

표 7 MNIST 및 CIFAR-10 데이터 사용시에 8개 최적화 알고리즘의 수렴 성능

Table 7 Convergence performance of 8 optimization algorithms for MNIST and CIFAR-10 data

optimizer	FNN MNIST		CNN		Total (4 cases)	
	2-layer	4-layer	MNIST	CIFAR-10	(Table 2~Table 3)	
	ranking	ranking	ranking	ranking	average ranking	ranking
SGD	8	3	6	6	5.75	7
Momentum	7	6	4	4	5.25	4
Nesterov	3	1	1	3	2.00	1
AdaGrad	5	5	8	5	5.75	7
RMSprop	1	8	5	8	5.50	5
Adam	6	7	7	2	5.50	5
KO	2	4	2	7	3.75	3
mSGD	4	2	3	1	2.50	2

8. 결론

본 논문에서는 머신러닝/딥러닝의 학습에 사용되는 8개의 최적화 알고리즘의 성능을 다수의 지역 최저점을 갖는 함수(데이터), MNIST 데이터, CIFAR-10 데이터의 세 가지 데이터에 대해서 실험적으로 비교해보았다. 이 세 가지 데이터에 대해서, 함수 2 경우(case), FNN 2 경우, CNN 2 경우, AE 1 경우의 총 7가지 경우에 대해, 학습 결과를 표 1~4에서 최적화 알고리즘별로 수렴 성능을 비교하였고, 표 2~4에서는 1,000 iteration 동안의 계산 시간을 제시하였다. 또 표 6에서는 MNIST 데이터, CIFAR-10 데이터 사용시에 FNN 2 경우, CNN 2 경우에 대해서 훈련 후에 테스트 이미지를 사용하여 분류 성능을 비교해보았다.

본 논문에서는 실험적인 성능 비교를 했으므로 표 2~4에 제시된 개개의 경우에 대한 순위는 절대적이지 않고, 수렴 성능 차이가 근소한 경우는 미니 배치의 특성상 실험에 따라 순위가 바뀔 수도 있다는 점을 참고하고 표를 봐야 한다. 그러나 표 5와 같이 7가지 경우(좌측) 또는 5개 경우(우측, 7가지 경우에서 함수 2 경우를 뺀 것), 표 6~7과 같이 4개 경우를 종합적(평균적)으로 보면 8개 알고리즘별로 수렴 성능과 분류 성능의 비교는 의미가 있다고 볼 수 있다.

AdaGrad는 손실함수의 경사가 큰 상태에서 시작한 경우에는 학습 초반에 학습이 중단되는 단점이 있는데, 그림 6과 그림 7에서 MNIST와 CIFAR-10 데이터를 CNN에 적용할 때 이 단점을 잘 보여주고 있다. 표 5의 오른쪽과 같이 MNIST 데이터와 CIFAR-10 데이터를 사용한 5개의 경우에 대해서는 AdaGrad가 종합순위 8순위로 수렴 성능이 가장 나쁘다. AdaGrad의 단점을 개선한 것이 RMSprop인데 표 5에서 RMSprop이 AdaGrad보다 수렴성능이 우수한 것을 볼 수 있다. mSGD는 AdaGrad와 RMSprop처럼 손실함수의 경사만 이용해서 가중치를 업데이트하는데 mSGD는 표 5에서 좌우측 모두 1순위를 하고 있어서 RMSprop보다 종합순위로 수렴 성능이

우수하다고 볼 수 있다.

Momentum(SGD momentum)은 SGD에서 손실함수의 경사도와 경사도의 관성을 함께 고려한 알고리즘인데, 표 5를 보면 수렴 성능이 거의 하위를 차지하고 있다. Momentum을 개선한 알고리즘인 Nesterov는 표 5에서 좌우측 모두 2위로 아주 우수한 수렴성능을 보여주었다. Adam은 많이 사용되는 최적화 알고리즘으로 Momentum과 RMSprop를 결합한 알고리즘이다. 그런데 Momentum의 성능이 안 좋아서 Adam의 성능은 종합적으로 표 5에서 좌우측 각각 6순위와 5순위를 차지하였다. 표 5로 유추한다면 Nesterov와 RMSprop를 결합한 알고리즘은 수렴 성능이 우수하리라고 예측할 수 있다.

KO 알고리즘은 표 5에서 좌우측 각각 4순위와 3순위를 차지하여 중간 정도의 수렴 성능을 보였는데, 칼만필터의 여러 파라미터는 3장의 함수 $f_1(x, y)$ 에 대해서 구한 값을 그 후에도 계속 사용하여서 원래 성능보다 저평가 되었다고 볼 수 있다.

8개 최적화 알고리즘의 계산 시간은, mSGD를 제외한 7개는 계산시간 차이가 적으므로 주로 SGD와 mSGD를 비교하였다. mSGD의 경우 3-point mSGD[9]를 사용하였는데 3번의 손실함수 값을 계산하기 때문에 계산 시간이 SGD의 3배 정도 소요될 것으로 예상했는데, 표 2에서 MNIST 데이터에 FNN 사용시에 2-layer일 때 62.6%, 4-layer일 때 50% 길었고, 표 4에서 AE에 FNN을 사용할 때는 68% 더 소요되었다. 표 3에서 MNIST 데이터에 CNN 사용시에는 7.3배, CIFAR-10 데이터에 CNN 사용 시에는 7.6배 더 소요되었다.

표 6~7에서는 MNIST 데이터, CIFAR-10 데이터 사용시에 FNN 2 경우, CNN 2 경우에 대해서 훈련 후에 테스트 이미지를 사용한 분류 성능과 훈련 시의 수렴 성능을 각각 보여주고 있다. 손실함수의 경사만을 이용한 SGD, AdaGrad, RMSprop가 경사의 관성을 같이 이용한 Momentum, Nesterov, Adam보다 손실함수의 수렴 성능과 분류 성능 모두 뒤지는 것을 보여주고 있다. 표 7에서 손실함수의 경사만을 이용한 mSGD는 분류 문제에서 수렴 성능은 종합 2위이지만 표 6에서 분류 성능은 4위로 Momentum, Nesterov, Adam보다 모두 뒤지는 것을 보여주었다. 표 6~7은 Nesterov가 분류 문제에서는 수렴 성능이나 분류 성능이 모두 1위임을 보여 주었다.

Acknowledgements

이 논문은 2024년도 중앙대학교 연구년 결과물로 제출됨.

(This research was supported by the Chung-Ang University research grant in 2024)

References

- [1] Y.-H. Ryu, D.-Y. Kim, D. Oualid and D.-J. Lee, "Deep learning-based lane recognition through longitudinal and lateral nonlinear model prediction control secure motion planning (written in Korean)," Proceedings of the 2022

- Information and Control Symposium, pp. 61-63, 2022.
<https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE11065764>
- [2] N. Fatima, "Enhancing Performance of a Deep Neural Network, A Comparative Analysis of Optimization Algorithms," *Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 9, no. 2, pp. 79-90, 2020.
 DOI:<https://dx.doi.org/10.14201/ADCAIJ2020927990>
- [3] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.
 DOI:10.48550/arXiv.1609.04747
- [4] A. Zhang, Z. C. Lipton and S. J. Smola, *Dive into Deep Learning*, pp. 473-555, 2022.
 DOI:10.1017/9781009389426
- [5] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145-151, 1999.
 DOI:10.1016/S0893-6080(98)00116-6
- [6] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, issue 7, pp. 2121-2159, 2011.
<https://jmlr.org/papers/volume12/duchil1a/duchil1a.pdf>
- [7] M. D. Zeiler, "AdaDelta: An Adaptive Learning Rate Method," *arXiv:1212.5701v1 [cs.LG]*, 2012.
 DOI:10.48550/arXiv.1212.5701
- [8] D. P. Kingma and j. L. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the 3rd International Conference on Learning Representations*, pp. 1-15, 2015.
 DOI:10.48550/arXiv.1412.6980
- [9] D.-S. Shim and J. Shim, "A Modified Stochastic Gradient Decent Optimization Algorithm with Random Learning Rate for Machine Learning and Deep Learning," *International Journal of Control, Automation, and Systems*, vol. 21, no. 11, pp. 3825-3831, 2023.
 DOI:10.1007/s12555-022-0947-1
- [10] X. Yang, "Kalman Optimization for Consistent Gradient Decent," *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3900-3904, 2021.
 DOI:10.1109/ICASSP39728.2021.9414588
- [11] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, The MIT Press, pp. 286-302, 2016.
 DOI:10.5555/3086952
- [12] G. Saito, *Deep Learning from Scratch* (written in Korean), Hanbit Media, inc., pp. 189-202, 2017.
<https://sdr1982.tistory.com/201>
- [13] F. Schneider, L. Balles and P. Heunnig, "DeepOBS: A Deep Learning Optimizer Benchmark Suite," *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, pp. 1-14, 2019.
 DOI:10.48550/arXiv.1903.05499
- [14] D. Choi, C.J. Shallue, Z. Nado, J.-H Lee, C.J. Maddison and G.E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," *arXiv:1910.05446v3 [cs.LG]*, 2020.
 DOI:10.48550/arXiv.1910.05446
- [15] A. Zohrevand and Z. Imani, "An Empirical Study of the Performance of Different Optimizers in the Deep Neural Network," *Proceedings of the 12th Iranian/Second International Conference on Machine Vision and Image Processing (MVIP)*, 2022.
 DOI:10.1109/MVIP53647.2022.9738743

저자소개



김규식(Kiu-Sik Kim)

2024년 2월 중앙대학교 전자전기공학부 졸업.
 2024년 3월~현재 중앙대학교 대학원 전자전기공학과 석사과정 재학중. 관심분야는 관성항법, 로봇 SLAM, 영상-관성 odometry, 인공지능



심덕선(Duk-Sun Shim)

1984년 2월, 1986년 2월 서울대학교 제어계측공학과 졸업(공학사, 공학석사), 1993년 12월 미시간대학교 항공우주공학과 졸업(공학박사), 1994년 1월~1995년 1월 미시간대학교 전기공학및컴퓨터과학과(EECS) 포스트닥, 1995년 3월~현재 중앙대학교 전자전기공학부 교수, 2009년~2011년 JEET 에디터, 2014년~2017년 IJCAS 에디터, 2017년 제어로봇시스템학회 펠로우 회원, 2018년 대한전기학회 정보및제어부문회장, 2018년, 2022년 대한전기학회 부회장. 관심분야는 강인제어, 위성항법, 관성항법, 로봇 SLAM, 영상-관성 odometry, 인공지능.